

## **Title: Squareston -- The Use of Coordinate Geometry and Mathematical Modeling in Urban Planning**

### **Brief Overview:**

This unit is for use with other subject matter in a geometry, computer science, or integrated mathematics course. It includes four closely related activities designed to familiarize the student with coordinate geometry while reviewing mean, median, and the x-y coordinate system. The activities use well defined, easy to understand problems that progress from a very simple problem in Activity 1 to relatively complex problems requiring computer support in Activity 4.

### **Links to NCTM 2000 Standards:**

- **Mathematics as Problem Solving, Reasoning and Proof, Communication, Connections, and Representation**

These five process standards are threads that integrate throughout the unit, although they may not be specifically addressed in the unit. They emphasize the need to help students develop the processes that are the major means for doing mathematics, thinking about mathematics, understanding mathematics, and communicating mathematics.

Throughout the unit students relate the mathematical “solutions” that they derive to the decisions being made by city planners. As a consequence, the students see the use -- and potential misuse -- of mathematical modeling in decision making.

- **Number and Operation**

Activities 1 through 3 reinforce familiar scalar operations. In Activity 4, students build data matrices and use them to develop solution sets.

- **Patterns, Functions, and Algebra**

The connections between algebra and geometry are made apparent to the students through use of the Cartesian Coordinate System in modeling geometric problems and the use of algebraic equations to solve for unknown coordinates.

- **Geometry and Spatial Sense**

Students use coordinate systems of varying scales to solve increasingly complex problems related to urban planning.

- **Data Analysis, Statistics, and Probability**

Students collect data from a coordinate system and calculate median and mean.

## **Links to Virginia High School Mathematics Core Learning Units:**

- **A.1**

The student will solve linear equations and inequalities in one variable, solve literal equations (formulas) for a given variable and apply these skills to solve practical problems. Graphing calculators will be used to confirm algebraic solutions.

- **A.2**

The student will represent verbal quantitative situations algebraically and evaluate these expressions for given replacement values of the variables. Students will choose an appropriate computational technique, such as mental mathematics, calculator, or paper and pencil.

- **A.3**

The student will justify steps used in simplifying expressions and solving equations and inequalities. Justifications will include the use of concrete objects, pictorial representations, and the properties of real numbers.

- **A.14**

The student will solve quadratic equations in one variable both algebraically and graphically. Graphing calculators will be used both as a primary tool in solving problems and to verify algebraic solutions.

- **G.2**

The student will use pictorial representations, including computer software and coordinate methods to solve problems involving symmetry and transformation. This will include using formulas for finding distance, midpoint, and slope; investigating and determining whether a figure is symmetric with respect to a line or a point; and determining whether a figure has been translated, reflected, or rotated.

- **G.7**

The student will solve practical problems involving right triangles by using the Pythagorean Theorem and its converse, properties of special right triangles, and right triangle trigonometry. Calculators will be used to solve problems and find decimal approximations for the solutions.

- **COM.1**

The student will describe the program development cycle: defining the problem, planning a solution, carrying out the plan, debugging the program, and providing program documentation.

- **COM.2**

The student will write program specifications that define the constraints of a given problem. These specifications include descriptions of pre-conditions, post-conditions, the desired output, analysis of the available input, and an indication as to whether or not the program is solvable under the given conditions.

- **COM.3**

The student will design a step-by-step plan (algorithm) to solve a given problem. The plan will be in the form of a program flowchart, pseudo code, and a hierarchy chart and/or data flow diagram.

- **COM.4**

The student will use operating system commands, which include creating a new file, opening an existing file, saving a file, making a printed copy (hard copy) of the file, and executing a program.

- **COM.5**

The student will divide a given problem into manageable sections (modules) by task and implement the solution. The modules will include an appropriate user-defined function, subroutines, and procedures. Enrichment topics can include user-defined libraries (units) and object-oriented programming.

- **COM.6**

The student will design and implement the input phase of a program, which will include designing screen layout and getting information into the program by way of user interaction, data statements (BASIC), and/or file input. The input phase also will include methods of filtering out invalid data (error trapping).

- **COM.7**

The student will design and implement the output phase of a computer program, which will include designing output layout, accessing a variety of output devices, using output statements, and labeling results.

- **COM.9**

The student will define simple variable data types that include integer, real (fixed and scientific notation), character, string, and Boolean.

- **COM.10**

The student will use appropriate variable data types, including integer, real (fixed and scientific notation), character, string, and Boolean. This will also include variables representing structured data types.

- **COM.12**

The student will translate a mathematical expression into a computer statement, which involves writing assignment statements and using the order of operations.

- **COM.13**

The student will select and implement built-in (library) functions in processing data, which include trigonometric functions, absolute value functions, random number functions, end of line, end of file, and string.

- **COM.14**

The student will implement conditional statements that include if/then, if/then/else, case statements, and Boolean logic.

- **COM.15**

The student will implement a loop, including iterative loops, pretest loops, and post-test loops. Other topics will include single entry point, single exit point, preconditions, post-conditions and loop invariance.

- **COM.16**

The student will select and implement appropriate data structures, including arrays (one-dimensional and/or multidimensional), files, and records. Implementation will include creating the data structure, putting information into the structure, and retrieving information from the structure.

- **COM.18**

The student will test a program using an appropriate set of data. The set of test data should be appropriate and complete for the type of program being tested.

- **COM.19**

The student will debug a program using appropriate techniques (e.g., appropriately placed controlled breaks, the printing of intermediate results, and other debugging tools available in the programming environment), and identify the difference between syntax errors and logic errors.

- **COM.20**

The student will properly document a program including the preconditions and post-conditions of program segments, input/output specifications, the step-by-step plan, the test data, a sample run, and the program listing with appropriately placed comments.

- **COM.21**

The student will design, write, test, debug, and document a complete structured program which requires the synthesis of many of the concepts contained in previous standards.

- **COM.23**

The student will solve mathematical problems using formulas, equations, and functions. Problems will include those related to geometry, business, and leisure (e.g., sports and recreational activities).

**Grade/Level:**

Grades 8 - 10, Activities 1 - 3

Grades 10 - 12, Activity 4

**Duration/Length:**

Activities 1 – 3, two 90 minute periods.

Activity 4, one 45 minute classroom introduction and four and one-half 90 minute labs.

**Prerequisite Knowledge:**

Students should have working knowledge of the following skills:

- Plotting points in a Cartesian Coordinate System
- Solving equations involving squares and square roots
- Calculating the mean and median of data sets
- Calculating distance with the Pythagorean Theorem
- Use of external files and basic programming structures (Activity 4)

**Objectives:**

Students will be able to:

- calculate the midpoint of a line segment.
- plot points in a coordinate system.
- calculate the distance between two points in a coordinate system.
- calculate the median and mean of a data set.
- describe in their own words how mathematical modeling can be used in decision making.
- given a criterion function, use a simple mathematical model to assist in decision making.
- define appropriate data structures (Activity 4).
- read and display external data files (Activity 4).
- design and implement computer program for use in urban planning (Activity 4).

**Materials/Resources/Printed Materials:**

- Scientific calculators for each student
- Transparencies of city grids (see attached)
- Activity sheets (see attached)
- Appropriate compiler for computer language used (Activity 4)
- Data files for Activity 4 labs (hardcopies provided with sample programs)

**Development/Procedures:**

The teacher should start with a brief review of the Cartesian coordinate system, the extent of which will depend on the capabilities of the students. Activity 1 provides practice/review in plotting coordinates and familiarizes students with the use of x-y coordinates to determine distance along gridlines. Activities 1 and 2 introduce the use of coordinate systems in mathematical modeling. Additionally, Activity 2 provides practice in calculating median and mean and relates these statistics to decision making.

Activity 3 emphasizes use of the Pythagorean Theorem (distance formula) to determine the distance between two points in a coordinate system. Activity 4 is an extensive series of computer labs that extend the basic concepts of the previous activities in more complex situations.

### Assessment:

Students are assessed on:

- Group interaction (teacher observations)
- Understanding and use of the coordinate system, capacity to calculate midpoints and distance in a coordinate system, and understanding and use of the statistics median and mean (written submissions)
- Use of mathematical models to aid decision making (oral presentations of their solutions)

<b><i>RUBRIC</i></b>	<b>3</b>	<b>2</b>	<b>1</b>
<b>Group Interaction</b>	High level of constructive participation	Medium level of constructive participation	Low level of constructive participation
<b>Analysis</b> Coordinates Midpoint Distance Median Mean	Proficient (Over 85% accurate)	Basic (Over 64% accurate)	Below Basic (Under 65% accurate)
<b>Synthesis</b> (Oral presentations)	Demonstrates high level of understanding	Demonstrates moderate level of understanding	Demonstrates low level of understanding

### Extension/Follow Up:

Activity 4 is a computer-based extension of Activities 1 – 3. Activity 3 lends itself to an extension using calculus to determine the point of minimum total distance. Activity 3 also lends itself to an extension wherein populations are considered to be uniformly distributed and the centroids of various shapes are used to characterize population centers.

**Authors:**

Sharon Benson  
Thomas Dale High School  
Chesterfield County Public Schools, VA

Charles W. Brewer  
Lake Braddock Secondary School  
Fairfax County Public Schools, VA

Michael Diffley  
Bell Multicultural High School  
District of Columbia Public Schools

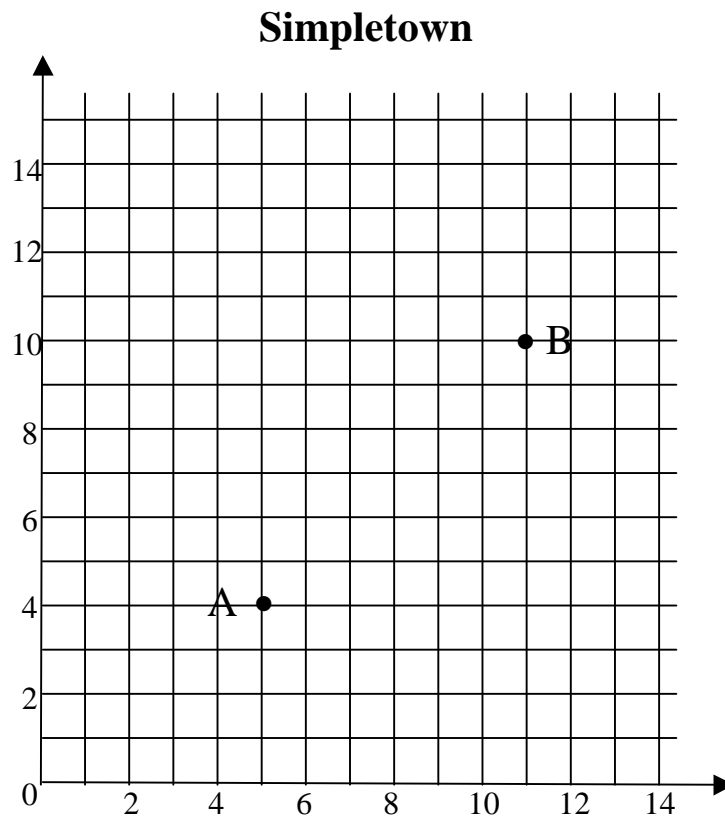
### Activity 1 - Simpletown

The following grid represents the streets and houses of Simpletown. The mayor is trying to determine the best place to build the rescue squad. This town has only two houses. On the town grid, the Adams are located at A(5,4) and the Browns are located at B(11,10).

**Objective:** Determine the “ideal” location for the rescue squad.

#### Assumptions

- grid lines represent two-way streets
- the paramedics can travel along existing streets directly to each house without obstacles
- each unit on the grid represents one town block
- buildings are located at intersections only



### *Questions*

1. Use the midpoint formula below to determine the location of the rescue squad such that the distance from the rescue squad to each house is the same.

$$M\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right) =$$

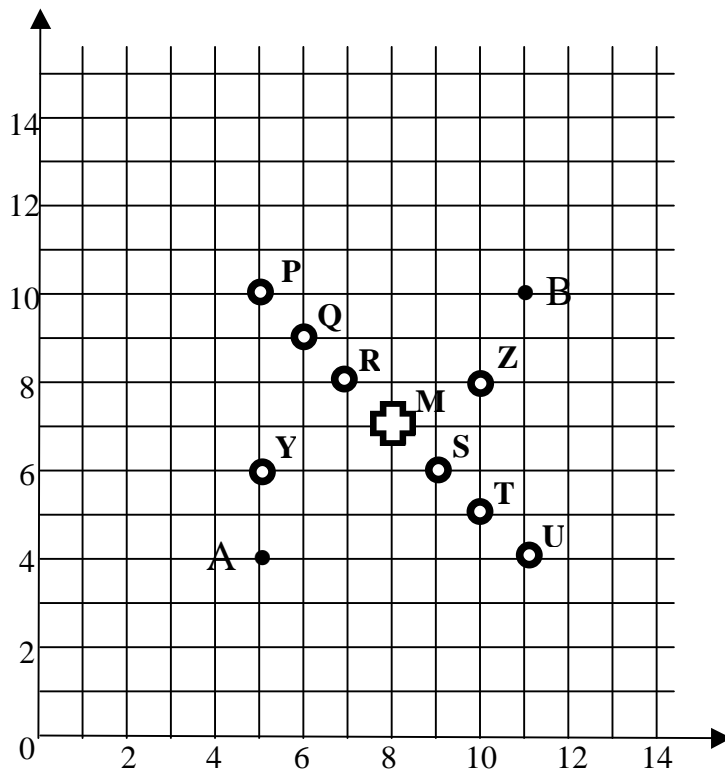
2. Plot point M on the grid and label its coordinates.
3. The distance between the rescue squad and the Adams' house is \_\_\_\_\_ blocks.  
The distance between the rescue squad and the Browns' house is \_\_\_\_\_ blocks.
4. Using what you discovered in problem #3, are there other possible locations for the rescue squad that are equidistant from both the Adams and the Browns? If so, list them here and label them on the grid. What would the Adams and Browns think about the fairness of any of these locations?
5. Suppose the criteria have changed from finding an equidistant location to finding a location that minimizes total distance. For example, if the rescue squad is located at point M(8,7), the total distance to the houses is 12 blocks ( $6 + 6 = 12$ ). If the rescue squad was located at the point with coordinates (5,6), total distance to each house is still 12 blocks ( $10 + 2 = 12$ ). What other locations minimize total distance? List them below and put a few on the grid.
6. What type of pattern exists with the locations that meet the new criteria?

7. Which criteria should be used in determining the “ideal” location for the rescue squad?  
Remember to consider what the individual families might think about fairness.
8. If there is more than one ideal location for the rescue squad, what might the mayor consider when deciding which location to select?

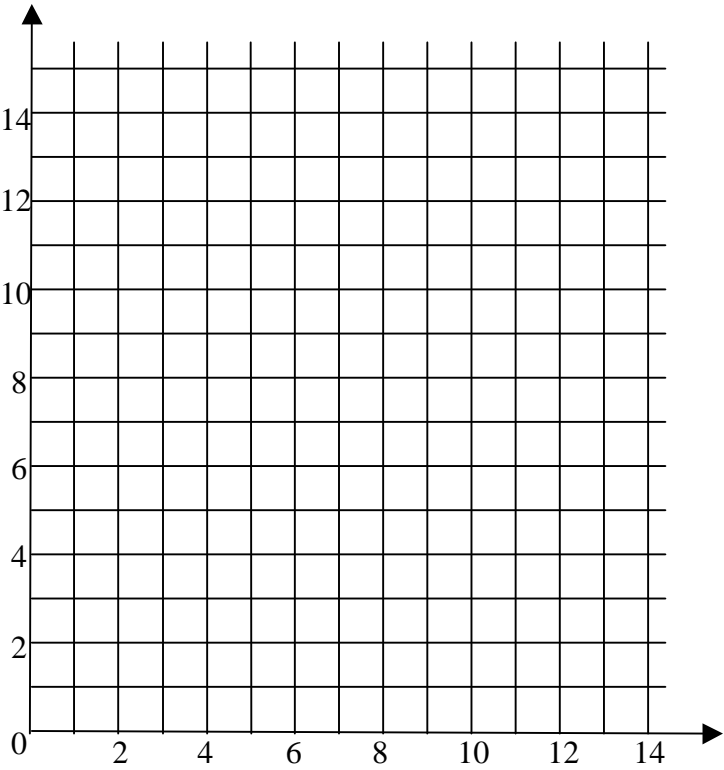
**Activity 1 – Simpletown**  
**Answer Key and Teacher's Notes**

1. M(8,7)
2. See answer grid below.
3. 6 blocks, 6 blocks
4. P(5,10)      Q(6,9)      R(7,8)      S(9,6)      T(10,5)      U(11,4)  
 Both families should think that any of these locations are fair.
5. and 6. Answers will vary. Any point within the 6X6 square formed with A and B as corners minimizes total distance. For example: Y(5,6) or Z(10,8).

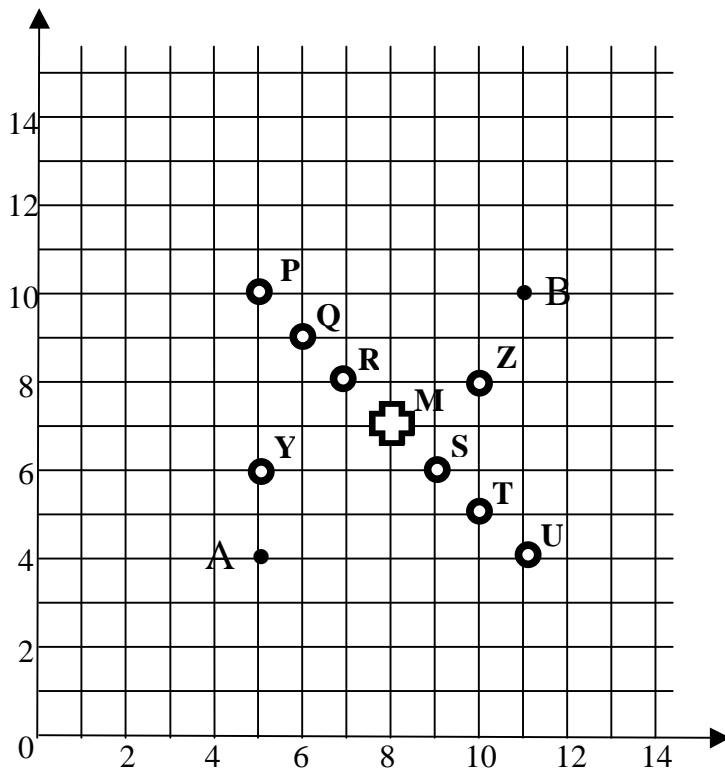
**Simpletown**



7. In this situation, the ideal location is the one that minimizes total distance *and* is equidistant from the two houses. The Browns probably would not be too happy about the rescue squad located at point Y.
8. Answers will vary.



Simpletown



## Activity 2 - Squareston

The following grid represents the streets of Squareston. The city council is trying to determine the best place to build the rescue squad. Houses, apartment buildings and businesses are represented by points A through E.

**Objective:** Determine the ideal location for the rescue squad.

### Assumptions

- grid lines represent two-way streets
- the paramedics can travel along existing streets directly to each house without obstacles
- each unit on the grid represents one city block
- buildings are located only at intersections

### Questions

1. Plot and label the following points on the city grid.

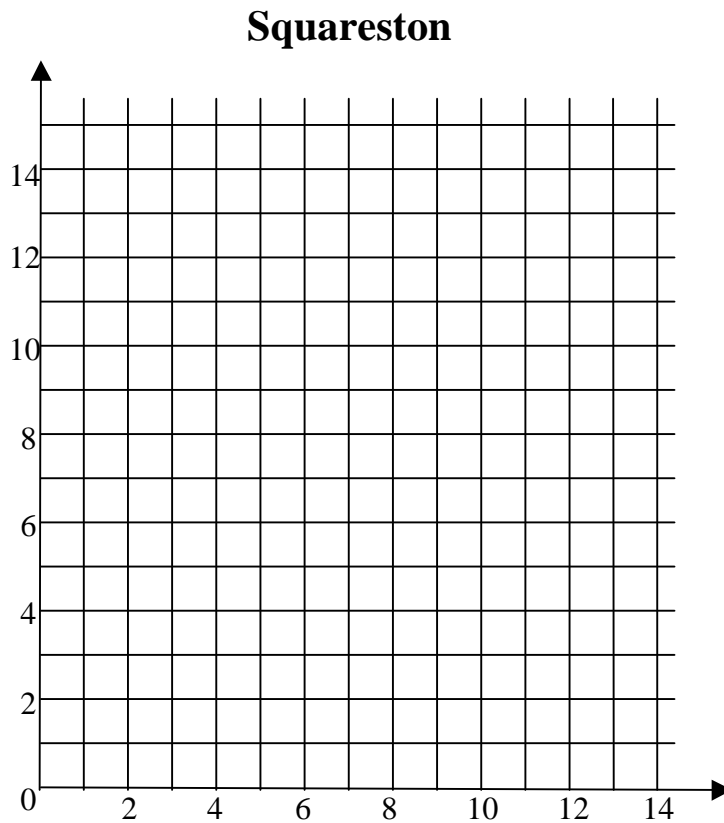
A( 2, 14 )

B( 1, 13 )

C( 4, 12 )

D( 3, 10 )

E( 14, 1 )



- In order to find the coordinates of the point with the minimum travel distance to each of Squareston's buildings, calculate the *arithmetic mean* of the x-coordinates and y-coordinates. On the grid, label these new coordinates R to represent the location of the rescue squad.

*mean of x-coordinates:*

*mean of y-coordinates:*

*location of rescue squad:* R(            )  
(round decimals to whole numbers since buildings can only exist at intersections)

- Determine the number of blocks between the rescue squad and each building. Complete the following table for point R.

**Distance in City Blocks**

	A	B	C	D	E	Total
Rescue Squad						
Point R						
Point S						

- What are some *advantages* to having the rescue squad located at point R?

What are some *disadvantages* to having the rescue squad located at point R?

5. Excluding point R, now find the *median* of the x-coordinates and the *median* of the y-coordinates of points A through E. On the grid, label these new coordinates S to represent another possible location for the rescue squad.

*median of x-coordinates:*

*median of y-coordinates:*

*location of rescue squad:* S(       )

6. Determine the number of blocks between the rescue squad and each building. Complete the table in question #3 for point S.
7. What are some *advantages* to having the rescue squad located at point S?

What are some *disadvantages* to having the rescue squad located at point S?

8. Is it better to use the mean or the median when determining the location of the rescue squad? Consider the total travel distance and the outlier point E. Write a paragraph explaining which location, R or S, is the best place to locate the rescue squad.

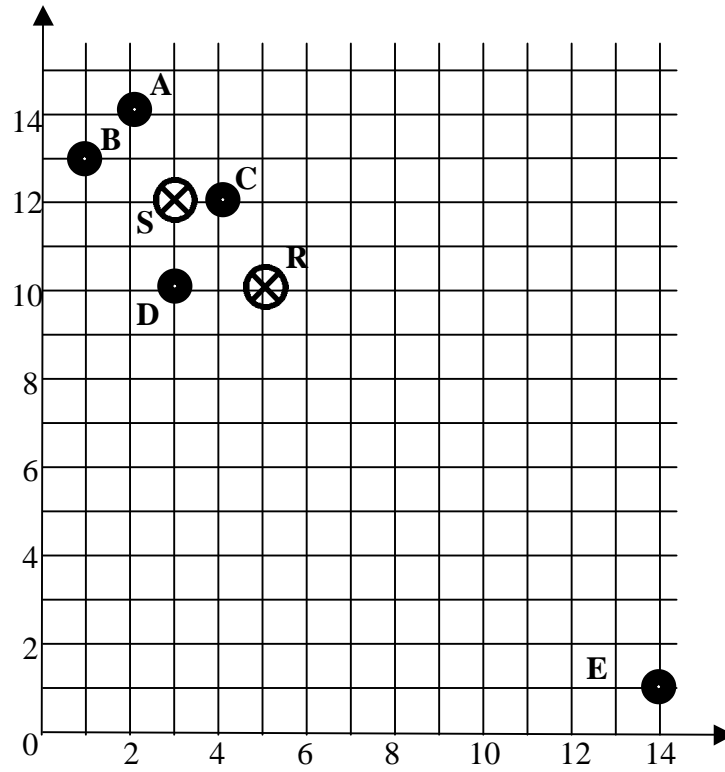
### *Consider This*

1. Activities 1 and 2 assumed the paramedics encountered no obstacles when driving along the streets. What are some realistic obstacles that emergency vehicles such as paramedics, firemen and police encounter when answering a call?
2. In Activities 1 and 2, deciding the location of the rescue squad was based on minimizing the travel distance between the rescue squad and the other points on the city grids. What other factors need to be considered besides the distance between points on the grids?
3. In Activity 2, points on Squareston's city grid represented residences and businesses. Do different buildings and their occupants have different potential rescue squad needs? For example, should the location of a storage warehouse be given the same consideration as the location of an apartment building?
4. How is distance between points affected when the paramedics do not have to travel along the grid lines? How would distance be calculated?

*Activity 2 – Squareston*  
*Answer Key and Teacher's Notes*

1.

**Squareston**



2. mean of x-coordinates:  $\frac{2 + 1 + 4 + 3 + 14}{5} = 4.8$

mean of y-coordinates:  $\frac{14 + 13 + 12 + 10 + 1}{5} = 10$

location of rescue squad: R( 5, 10 )

3. **Distance in City Blocks**

	A	B	C	D	E	Total
Rescue Squad						
Point R	7	7	3	2	18	37
Point S	3	3	1	2	22	31

4. See question #8.

5. *median of x-coordinates:* 2, 1, 4, 3, 14  $\rightarrow$  1, 2, 3, 4, 14

*median of y-coordinates:* 14, 13, 12, 10, 1  $\rightarrow$  1, 10, 12, 13, 14

*location of rescue squad:* S( 3, 12 )

**note:** Remind students that the data must be in numerical order before the median value is selected.

6. See table in question #3.

7. See question #8.

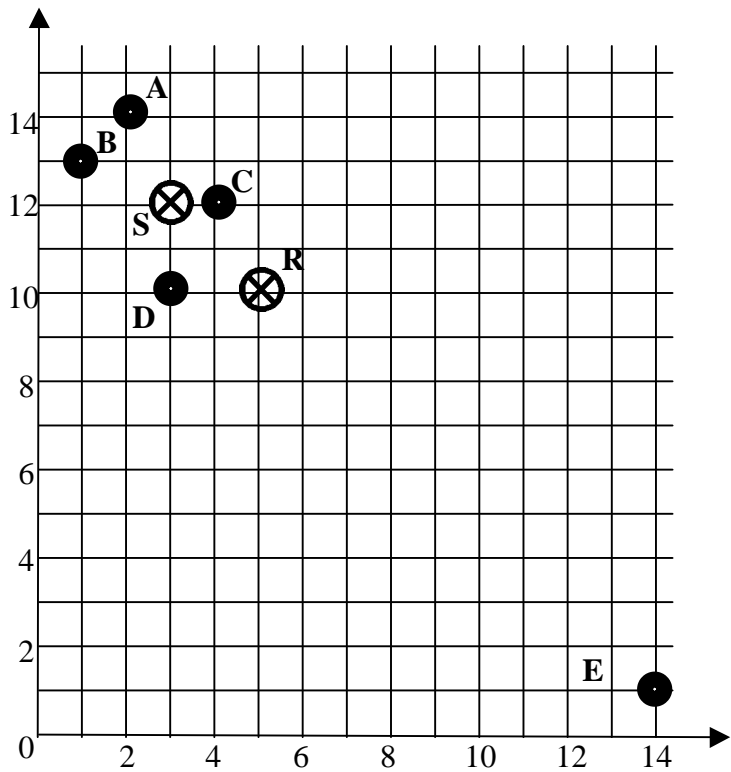
8. The purpose of calculating the mean and median of the x and y coordinates is to illustrate the effect that an outlier has on averaging data. The location of point S minimizes total distance and appears to be more advantageous. Point S is in the middle of the cluster of points A, B, C and D and 22 blocks away from point E. Point R is further away from points A and B and only 4 blocks closer to the outlier point E. Ask students which scenario seems best. Note that arguments may be presented for both points R and S. E is a point on a graph, but it also represents real people.

***Consider This***  
***Teacher's Notes***

These questions are designed to prompt the students to think about the same activities, but with different assumptions. It also designed to prepare them for Activity 3.

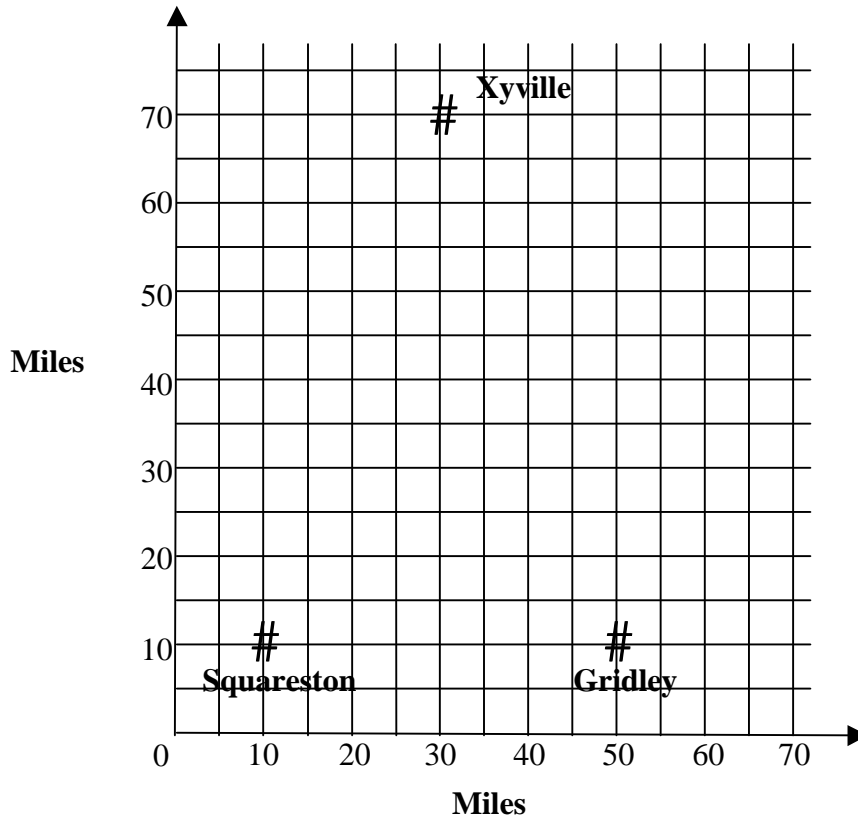
1. Answers will vary; however, realistic obstacles may be heavy traffic, weather conditions, clear directions to location, one-way streets etc.
2. Answers will vary. Besides distance, emergency personnel are concerned about getting to their destination in the shortest amount of time. How are calls prioritized if more than one call is received?
3. Answers will vary.
4. If paramedics could travel diagonally, distance between two points is truly minimized. In Activity #3, students will need to calculate the distance between two points on a coordinate plane using the distance formula.

Squareston



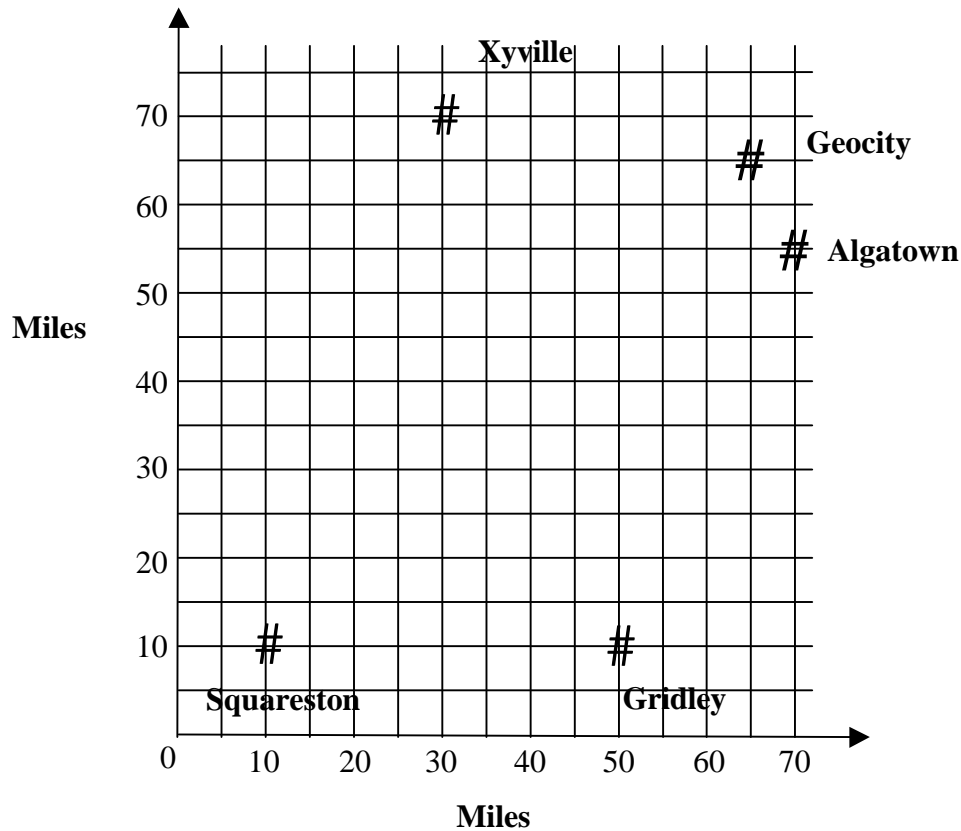
### Activity 3 Aero-Ambulance

Squareston is exploring the possibility of joining with two other cities, Gridley and Xyville, in establishing an aerial medical evacuation (medevac) service that would serve the three cities. (See the figure below for the relative locations of the three cities.)



1. If we use the same “minimum total distance” approach that we used to locate the rescue squad in Squareston, where would we locate the medevac service if helicopters were required to move along grid lines? What is the minimum total distance to the three cities from this point?
2. In reality helicopters are not required to fly along grid lines. Instead, they can travel in straight lines to each city. Find a location for the medevac service that reduces the total distance to all cities from what you found in problem #1 above. (Hint: you may want to look at points north of (30, 10) along the line  $x = 30$ .)
3. Minimum total distance is not the only criterion that we could use in selecting a location for the medevac service. If the mayors of the three cities agreed to locate the service at a point that was equidistant from each city, where would that point be?

Geocity and Algotown are also interested in joining the medevac consortium. (See the figure below for their locations.) The other mayors are happy to hear this since these two new cities will help share the costs.



1. Determine the location for the medevac station using:
  - The median x and y coordinates of the cities.
  - The mean x and y coordinates of the cities.
2. The mayor of Squareston is not happy with either “solution”. Could we solve this problem by identifying a point that is equidistant from all five cities?

3. Because Squareston is the largest of the five municipalities, the mayor of Squareston states that any decision must account for populations. We can do this by weighting (multiplying) the x and y coordinates of each city by their population and dividing the sum of these weighted coordinates by the total population of all the cities. Complete the chart below to determine the location of the medevac station.

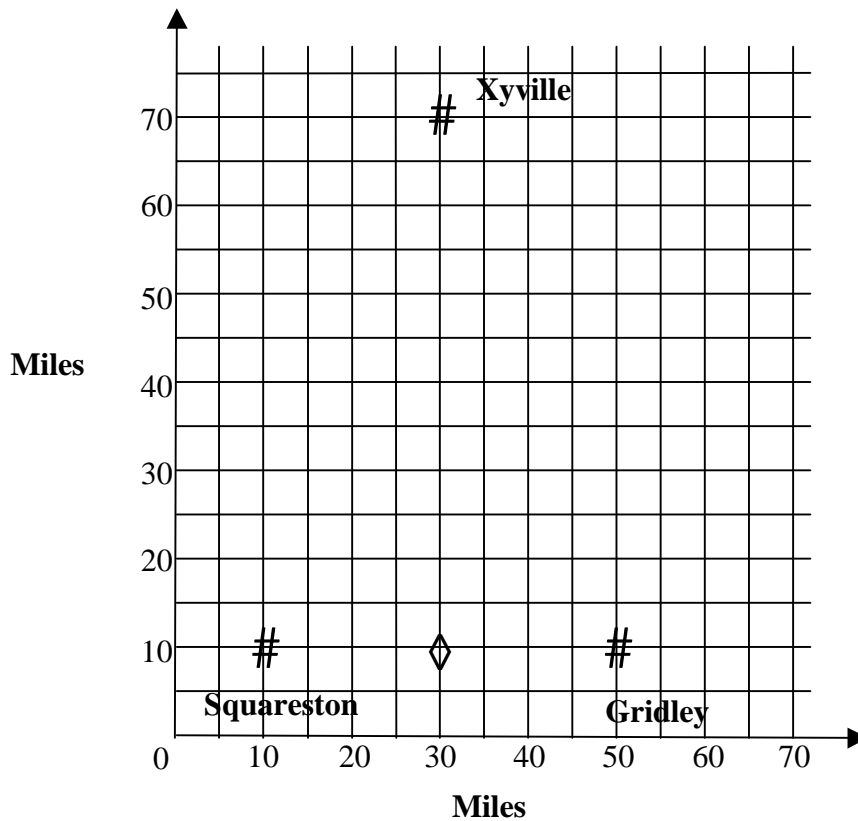
A City	B Population	C x	D y	E weighted x (B x C)	F weighted y (B x D)
Algatown	1,000	70	55	70,000	55,000
Geocity	1,000				
Gridley	2,000				
Squareston	4,000				
Xyville	2,000				
<b>Sum</b>	<b>10,000</b>				

4. From this location, how far is it to Squarestown? How far is it to the town farthest from the medevac station? Is this a reasonable solution to the problem?

## Teacher's Notes

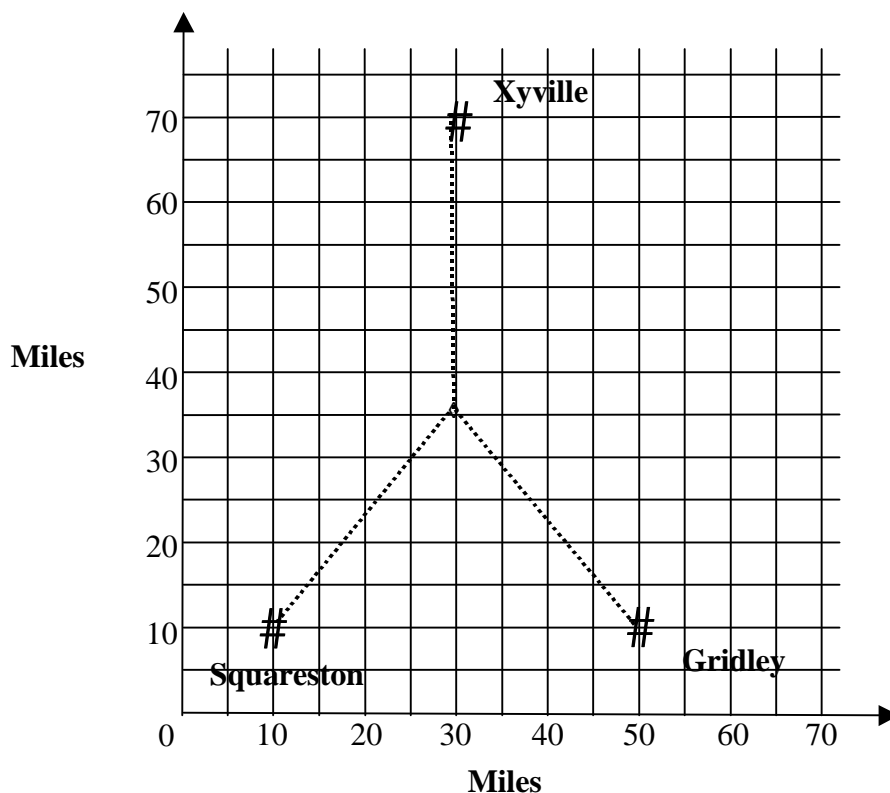
### Activity 3 Aero-Ambulance

Squareston is exploring the possibility of joining with two other cities, Gridley and Xyville, in establishing an aerial medical evacuation (medevac) service that would serve the three cities. (See the figure below for the relative locations of the three cities.)



1. If we use the same “minimum total distance” approach that we used to locate the rescue squad in Squareston, where would we locate the medevac service if helicopters were required to move along grid lines? What is the minimum total distance to the three cities from this point?

**Coordinates of the point with minimum total distance to all three cities travelling along gridlines are (30,10), the median of the x and y coordinates. From this point, the minimum total distance is 100 miles. (See above.)**



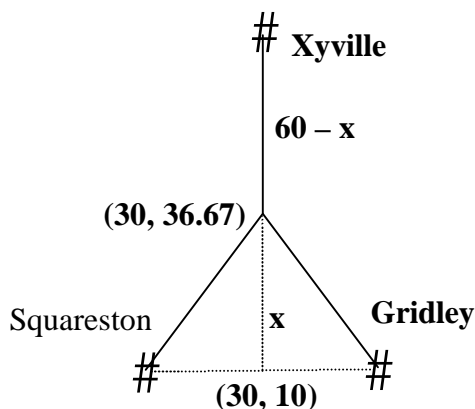
2. In reality helicopters are not required to fly along grid lines. Instead, they can travel in straight lines to each city. Find a location for the medevac service that reduces the total distance to all cities from what you found in problem #1 above. (Hint: you may want to look at points north of (30, 10) along the line  $x = 30$ .)

If students try locations farther north along the line  $x = 30$  miles, and use the distance formula (Pythagorean Theorem), they will discover that the total distance decreases at first and then begins to increase:  $y = 15$ ,  $D_t = 96.23$ ;  $y = 20$ ,  $D_t = 94.72$ ;  $y = 25$ ,  $D_t = 95$ ;  $y = 30$ ,  $D_t = 96.57$ ;  $y = 40$ ,  $D_t = 102.11$ .

$D_t(\min)$  occurs at  $y = 21.54$  and equals 94.64 although proof of this will be beyond the reach of students who have not taken advanced trigonometry or calculus.

3. Minimum total distance is not the only criterion that we could use in selecting a location for the medevac service. If the mayors of the three cities agreed to locate the service at a point that was equidistant from each city, where would that point be?

This question is designed to have the students “refocus” on the intent of the exercise. Hopefully, they will arrive at the need for an equitable solution.



#### Equidistant Solution

$$60 - x = \sqrt{x^2 + 20^2}$$

$$x^2 - 120x + 3600 = x^2 + 400$$

$$120x = 3200$$

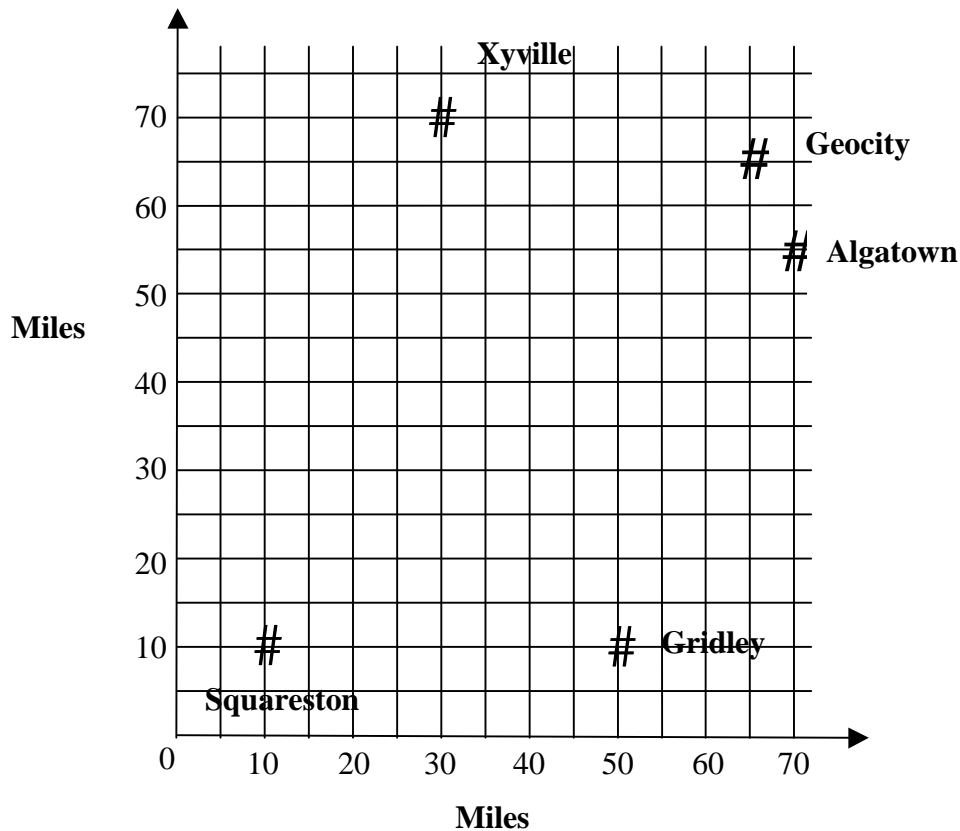
$$x = 26.67$$

Coordinates of equidistant point: (30, 36.67)

From this point, all cities are 33.33 miles from the medevac station.

Although the above closed solution will be within the capabilities of some students, iterative solutions (guess and check) will be a popular solution mechanism.

Geocity and Algatown are also interested in joining the medevac consortium. (See the figure below for their locations.) The other mayors are happy to hear this since these two new cities will help share the costs.



- Determine the location for the medevac station using:
  - The median x and y coordinates of the cities. **(50, 55)**
  - The mean x and y coordinates of the cities. **(45, 42)**
- The mayor of Squareston is not happy with either “solution”. Could we solve this problem by identifying a point that is equidistant from all five cities?

**There is no point equidistant from all five cities since they do not lie on a circle.**

3. Because Squareston is the largest of the five municipalities, the mayor of Squareston insists that any decision must account for populations. We can do this by weighting (multiplying) the x and y coordinates of each city by their population and dividing the sum of these weighted coordinates by the total population of all the cities. Complete the chart below to determine the location of the medevac station.

A City	B Population	C x	D y	E weighted x (B x C)	F weighted y (B x D)
Algatown	1,000	70	55	70,000	55,000
Geocity	1,000	<b>65</b>	<b>65</b>	<b>65,000</b>	<b>65,000</b>
Gridley	2,000	<b>50</b>	<b>10</b>	<b>100,000</b>	<b>20,000</b>
Squareston	4,000	<b>10</b>	<b>10</b>	<b>40,000</b>	<b>40,000</b>
Xyville	2,000	<b>30</b>	<b>70</b>	<b>60,000</b>	<b>140,000</b>
<b>Sum</b>	<b>10,000</b>	<b>NA</b>	<b>NA</b>	<b>335,000</b>	<b>320,000</b>

**From the chart above, the weighted average of the coordinates are: (33.5, 32).**

4. From this location, how far is it to Squaretown? How far is it to the town farthest from the medevac station? Is this a reasonable solution to the problem?

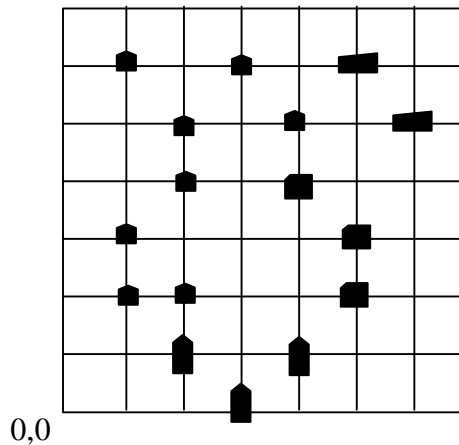
**Distance to Squareston is 32.2. Distance to Geocity is 45.6. Given Squareston's larger population, this would appear to be a reasonable solution.**

## Activity 4

### Squareston City Planning

**Introduction** The location of public facilities in a city is determined by many factors. In an ideal case, facilities are located to provide the “best” service to the most people. In reality, however, factors such as availability of land, costs, or politics, may produce a less than ideal solution. “Best” service may mean minimizing response times for the fire department and ambulance services or the convenient location of libraries and recreation facilities.

In this series of labs, you will develop data structures to model an orderly city (Squareston) and use those data structures to make decisions concerning the location of the public facilities. To simplify the problem, you may assume that the streets of Squareston are laid out on a grid pattern as shown to the right with buildings located only at the intersections. Not all intersections have buildings and existing buildings may be of different types. Four types of buildings will be considered:



■ Single-family dwellings

■ Multi-family dwellings

■ Industrial facilities

■ Stores

The criteria for determining the ideal location of the facility will depend upon the type of facility; therefore the data structure you develop to describe a building should provide all the data necessary to satisfy the differing criteria. For example, when considering the placement of libraries and recreational facilities, the number of families at a location is important while the location of stores or factories are not primary factors. The location of a fire house or rescue squad should consider the location of industrial facilities and stores however.

Complete the following:

1. What data should be stored for each building?
2. Define a data structure to store this data.
3. What data should be stored for the city?
4. Define a data structure to store this data.

## Lab 1A – Reading and displaying the Squareston data.

**Introduction** In this lab, you will create the data structures necessary to model the buildings and the city of Squareston, fill those data structures with the appropriate data, and display maps of Squareston to demonstrate that you have been able to do so.

### **Requirements**

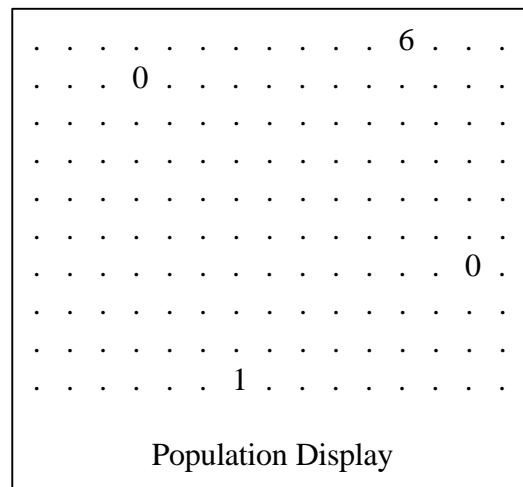
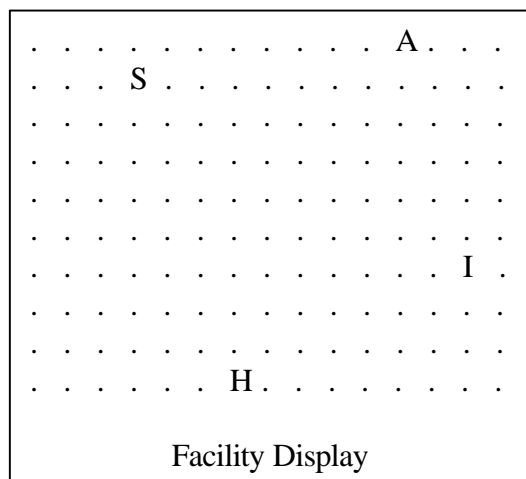
Using good design techniques, create a program that will read the data file and display the data on the screen. Intersections without facilities should be displayed with a dot (.). Use data file, citya.dat. The format of the file is as follows:

- First, two integers define the number of rows and columns in the grid describing Squareston.
- Then, groups of five fields describe individual facilities:
  - integer – row number
  - integer – column number
  - char – type of facility (H – single-family, A – multi-family, I – industrial, S – store, L – library, F – firehouse.)
  - integer – number of families
  - double – fire ranking

Example of a data file: (Note that facilities may be entered in random order.)

```
10 15
0 6 H 1 1
9 12 A 6 2
4 13 I 0 5
8 3 S 0 0.7
```

The data above would look like this: ( Lower left corner is 0, 0, but output starts at upper left corner. )



### Lab 1B – Squareston Library

The citizens of Squareston wish to construct a library that will be convenient to the most number of families. “Convenient” is defined as shortest distance along the existing streets. In locating a library or recreational facility, the population density, i.e., number of families, at a location must be considered. Stores and industrial facilities are not significant factors in determining the location unless they occupy the desired location.

Using good program design, create a program that will:

- Read and display the data for Squareston contained in data file, cityb.dat.
- Given a proposed library location (user keyboard input), determine:
  - The longest distance a family is located from the library.
  - The shortest distance a family is located from the library.
  - The average distance families are located from the library.
  - The median distance families are located from the library.

## Lab 1C – Squareston Library – Ideal location

**Introduction** There are many possible locations that might be considered for the library. In this lab, you will modify and extend your previously written program to iteratively examine all possible locations and identify the “best” location or locations for the library.

Additional design factors/ questions:

- Does the requirement, convenient to the most number of families, imply finding the mean or median of the distances from the library to various dwellings?
- Is the distance from a multi-family dwelling more, less, or of the same significance as the distance from a single-family dwelling?
- Is there more than one possible location?

### **Requirements**

- Examine each possible location for possible placement of the library. Design a data structure to store/manage the data, such as longest distance, shortest distance, mean distance, and median distance, produced by each examination.
- Compare this data to produce a recommendation(s) as to location of the new library.
- Display the map of Squareston to show the recommended location.
- Describe below the factors used to determine the recommended location and how you might choose between locations that seem equally acceptable.

---

---

---

---

---

---

---

---

---

---

Enrichment:

Modify the program to facilitate the addition of new buildings to your layout and to store this data in the data file.

## Lab 2 – Squareston Firehouse

In addition to the new library, the citizens of Squareston have decided that a facility is required to house the newly formed fire department. Travel time from the firehouse to possible fire locations is of major concern in selecting the location. Also, the industrial facilities present a significant hazard to the city because an uncontrolled fire could produce toxic fumes and cause major collateral damage to the rest of the city. The businesses, however, all have sprinkler systems installed.

1. In picking the location of the firehouse, what criteria need to be considered?
2. What modifications (if any) to the program from Lab 1 are required to find the ideal location of the firehouse?

### **Requirements**

- Implement any necessary modifications to your program from Lab 1 and execute using data file, city2.dat, to determine:
  - The ideal location(s) for the firehouse.
  - The longest distance a facility is located from the firehouse.
  - The shortest distance a facility is located from the firehouse.
  - The average distance facilities are located from the firehouse.
  - The median distance facilities are located from the firehouse.
- As in Lab 1, display the layout of Squareston using simple text output.

Enrichment. The industrial facilities present such a significant problem that a separate firehouse will be built to service these facilities. Modify your program to determine the “ideal” location of this firehouse and of the firehouse to service the rest of the city.

Describe below, when separate firehouses would or would not be a good use of resources.

---

---

---

---

---

---

## **Activity 4 – Teacher Notes**

This group of classroom and lab exercises is designed to review apvector, apmatrix, loops, conditionals, math functions, and external files. These exercises also provide the opportunity to introduce and use structs to group dissimilar data.

References and suggested reading assignments from “C++ for You”, AP Edition, by Maria and Gary Litvin, Skylight Publishing.

Apvectors and apmatrices – Chapter 5 (Review)

Conditionals – Chapter 6 (Review)

Loops and Iterative Functions – Chapter 7 (Review)

Input/Output – Appendix C (Review)

Structures – Chapter 13 (May be new material)

## **Squareston City Planning – Classroom Discussion – one regular (47 min) period**

Preparatory Assignment – Read Chapter 13, “C++ for You++”

### **Objectives:**

- The four questions are designed to lead to a discussion of how to store dissimilar data and the introduction of structs and their use. Some students may suggest the use of parallel vectors or matrices to store the data. However, the number of data items required for each facility makes this an unwieldy approach. Similarly, when data is calculated in the latter lab exercises, there are multiple elements that will need to be stored in each element of an apmatrix. Data elements that need to be stored are type of facility, number of families, and a fire ranking (a measure of how significant a fire risk the facility poses). The location coordinates could be stored but can also be determined from the city data structure below.
- Once the concept of a struct as a data structure for an individual facility is understood by the student, a data structure to describe the city should be easily visualized as an apmatrix of these structs.

Homework assignment: Preliminary design of Lab 1A.

### Lab1A – Reading and Displaying the Sqaureston Data – One 90 min block

**Objectives:** This lab will provide a review of techniques of reading external files and use of apmatrices. Students will use an apmatrix of structs to store data.

- Students should implement a struct to store data for each facility. Recommend that students include a constructor to initialize data elements appropriately. (See sample program.)
- Given the example data file, students should write an input function that first extracts the row and column information for the apmatrix representing the city. Then, using a loop that tests for end of file or failure to extract data, the function should extract the five items of data for each facility stored in the file.
- Students should write a map display function to display the data inputted. Two versions of a map as shown on the lab sheet. Either or both can easily be generated using nested for loops. The only trick is to remember that the first row printed is the last row in the apmatrix, i.e., the row for loop goes for last to first while the column for loop goes from first to last.

Homework Assignment: Preliminary design Lab 1B

### Lab 1B – Sqaureston Library – One 90 min block

**Objectives:** This lab requires the student to create functions that will determine the distances from a designated location to each of the housing facilities and then determine the mean, median, longest, and shortest of these distances. Good program design and proper use of functions and parameter passing should be emphasized.

- Students should design function to the distance along the grid between the designated location and the facility being examined. Use of absolute value function from math.h may be required. Note the distance is the sum of the horizontal and vertical coordinate differences, not the direct route. These distances should be stored in an appropriate data structure (apvector) and passed as reference parameter to facilitate further comparisons and calculations. One additional factor that must be considered in this function is the weighting of multi-family dwellings. If an apartment has four families, vice one in a house then it should be considered four times in the calculation. For mean calculations, the distance can merely be multiplied by four, but for median calculations it is more appropriate to have four entries of the distance.
- Students should design appropriate functions to calculate mean and median distances. A sort function will be required.
- In all of the functions, the proper use of structs will be required.

Homework Assignment: Preliminary design of Lab 1C

### Lab 1C – Ideal Library Location – One and one-half 90 min blocks

**Objectives:** This lab builds on the previous two and requires the student to make and justify design decisions on how to “best” locate a library.

- Students should create an iterative function to examine each possible location to determine the mean, median, longest, and shortest distances associated with that location. An apmatrix of structs could be used to store the data.
- From this data, the program should generate a recommended location for the library. Possible criteria include the smallest mean or smallest median distance from the library. The latter will likely give fewer discrete solutions than the former, but is not necessarily the “best” answer. Data file, cityx.dat, contains outliers that will demonstrate difference between “equal pain” (mean) and “shortest distance” (median) solutions.
- Program should incorporate recommended location into city data structure and display new map.
- Students should justify design decisions in short paragraph.

Homework Assignment: Enrichment activity or preliminary design of Lab 2

Enrichment activity (extra credit). Add a function to input facility data from keyboard and another function to write data back to data file.

### Lab 2 – Squareston Firehouse – One 90 min block

**Objective:** This lab is a modification of previous lab to reflect a change in selection criteria.

- Student may opt to modify the function used to determine distances for locating library (weighted by population) or write a similar function weighted by the fire ranking. Rest of the lab is essentially the same as Lab 1.

Enrichment: The addition of a separate firehouse to service the industrial facilities will require functions that select only industrial facilities and other than industrial in doing the calculations. An examination of when this is a reasonable plan might also be included.

```

// Lab 1A

#include <iostream.h>
#include <fstream.h>
#include <apmatrix.h>
#include <conio.h>

struct facility
{
    char type;          // H-single-family, A-multi-family, I-factory, S-store
    int num;            // number of families, 0 for factories and stores
    double fireRanking; //Relative measure of fire fighting requirements
    facility():type('.'),num(0),fireRanking(0){};          // constructor
};

typedef apmatrix<facility> city;

void getMap (city &);
//Pre:  City object is declared.
//Post: Data file is read and stored in city.

void displayMap (const city &);
//Pre:  City object is defined.
//Post: Map of facilities in city is displayed.

int main()

{

    city squareston;

    getMap(squareston);

    displayMap (squareston);

    return 0;

}

void getMap (city &two)
//Pre:  City is defined.
//Post: Data file is read and stored in city.
{
    ifstream inFile ("citya.dat");
    if (!inFile)
    {
        cerr << "File is not found.";
        exit (-1);
    }
    int rows, cols, r, c;
    inFile >> rows >> cols;

```

```

    two.resize (rows, cols);
    while (inFile >> r >> c)
        inFile >> two[r][c].type >> two[r][c].num
            >> two[r][c].fireRanking;
}

void displayMap (const city &one)

//Pre:  City is defined.

//Post: Map of facilities in city is displayed.
{
    int rows = one.numrows();

    int cols = one.numcols();
    for(int r = rows -1; r>=0; r--)          // output last row first
    {
        for (int c = 0; c<cols; c++)          // outputs facility type
            cout << one[r][c].type << " ";
        cout<<" ";

        for (int c = 0; c<cols; c++)          // outputs number of families
            if (one[r][c].num == 0)
                cout << ". ";
            else cout << one[r][c].num << " ";
        cout << endl;
    }
}

/*citya.dat    Data file

8 8
1 1 H 1 1
1 3 H 1 1
1 5 F 0 4
2 2 H 1 1
2 4 H 1 1
2 6 F 0 6
3 2 H 1 1
3 4 S 0 0.7
4 1 H 1 1
4 5 S 0 0.7
5 1 H 1 1
5 2 H 1 1
6 2 A 3 1
6 4 A 5 2
2 3 A 4 1.5

```

## Output

.	.	.	.	.	.	.	.	.	.
.	.	A	.	A	.	.	.	.	.
.	H	H	.	.	.	.	.	.	.
.	H	.	.	.	S	.	.	.	.
.	.	H	.	S	.	.	.	.	.
.	.	H	A	H	.	F	.	.	.
.	H	.	H	.	F	.	.	.	.
.	.	.	.	.	.	.	.	.	.

.	.	.	.	.	.	.	.	.	.
.	.	3	.	5	.	.	.	.	.
.	1	1	.	.	.	.	.	.	.
.	1	.	.	.	.	.	.	.	.
.	.	1	.	.	.	.	.	.	.
.	.	1	4	1	.	.	.	.	.
.	1	.	1	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

\*/

```

// Lab 1B
#include <iostream.h>
#include <fstream.h>
#include <apmatrix.h>
#include <math.h>
#include <conio.h>
#include <iomanip.h>

struct facility
{
    char type; // H - single-family, A - multi-family, I - factory, S - store
    int num;    // number of families, 0 for factories and stores
    double fireRanking; // Relative measure of fire fighting requirements
    facility():type('.'),num(0),fireRanking(0){};
};

typedef apmatrix<facility> city;
typedef apvector<double> distance;

void displayMap (const city &);
//Pre:   City object is defined.
//Post:  Map of facilities and map of population displayed.

void getMap (city &);

//Pre:   City object is defined.

//Post:  Data in file cityb.dat stored in city object.

void getLocation (int &, int &, const city &);
//Pre:   City object is defined. ints are declared.
//Post:  User inputted location stored in the ints.

void measure(const city &, distance&, int, int);

//Pre:   City object and ints are defined. Distance object is declared.

//Post:  The distance of each dwelling from location passed via ints is
//       determined and stored in distance object. Multi-family dwellings
//       are weighted according to number of families.

void sort (distance &);

//Pre:   Distance object is defined and full.
//Post:  Distance object is sorted in ascending order.

void swap (double &, double &);

//Pre:   The doubles are defined.
//Post:  The values in the doubles are swapped.

double mean (const distance &);
//Pre:   Distance object is defined.
//Post:  The mean of the elements of distance object is returned.

double median (const distance &);

```

```
//Pre: Distance object is defined.  
//Post: The median of the elements of distance object is returned.
```

```
void outputData (distance &);
```

```
//Pre: Distance object is defined and full.  
//Post: Data in distance object is displayed.
```

```
int main()
```

```
{
```

```
    city squareston;
```

```
    int row, col;
```

```
    distance miles;
```

```
    getMap(squareston);
```

```
    displayMap (squareston);
```

```
    getLocation (row, col, squareston);
```

```
    measure(squareston, miles, row, col);
```

```
    outputData (miles);
```

```
    getch();
```

```
    return 0;
```

```
}
```

```
void getMap (city &two)
```

```
//Pre: City is defined.
```

```
//Post: Data in file cityb.dat stored in city.
```

```
{
```

```
    ifstream inFile ("cityb.dat");
```

```
    if (!inFile)
```

```
    {
```

```
        cerr << "File is not found.";
```

```
        exit (-1);
```

```
    }
```

```
    int rows, cols, r, c;
```

```
    inFile >> rows >> cols;
```

```
    two.resize (rows, cols);
```

```
    while (inFile >> r >> c)
```

```
        inFile >> two[r][c].type >> two[r][c].num
```

```
            >> two[r][c].fireRanking;
```

```
}
```

```
void displayMap (const city &one)
```

```
//Pre: City is defined
```

```
//Post: Map of facilities and map of population displayed
```

```
{
```

```
    int rows = one.numrows();
```

```
    int cols = one.numcols();
```

```
    for(int r = rows -1; r>=0; r--)
```

```
{
    for (int c = 0; c<cols; c++)
        cout << one[r][c].type << " ";
    cout<<" ";
    for (int c = 0; c<cols; c++)
        if (one[r][c].num == 0)
            cout << ". ";
        else cout << one[r][c].num << " ";

    cout << endl;
}
}
```

```

void getLocation(int &row, int &col, const city &one)
//Pre: City is defined. row and col are declared.
//Post: User inputted location stored in row and col.
{
    do
        {cout << endl <<"Enter the x and y coordinates of the proposed location:
";
        cin >> row >> col;
        }
    while(row < 0 || row > (one.numrows()-1) || col < 0 || col > (one.numcols()-1));
}

void measure (const city &three, distance &four, int x, int y)
//Pre: City, x, and y are defined. four is declared.
//Post: The distance of each dwelling from x and y is determined and
//      stored in four. Multi-family dwellings are weighted according to
//      number of families.
{
    int rows = three.numrows();
    int cols = three.numcols();
    four.resize (rows*cols);
    int num(0);
    for(int r = 0; r<rows; r++)
    {
        for (int c = 0; c<cols; c++)
            if (three[r][c].type != '.' && three[r][c].num!= 0)
                for (int i = 0; i<three[r][c].num; i++)
                    {four[num]= fabs(r-x)+fabs(c-y);
                    num++;
                    }
    }
    four.resize(num);
}

void sort(distance &six)
//Pre: Six is defined and full.
//Post: Six is sorted in ascending order.
{
    int smallest, items = six.length();
    for (int outer(0); outer < items - 1; outer++)
    {
        smallest = outer;
        for (int inner = outer + 1; inner < items; inner++)
            if (six[inner] < six[smallest])
                smallest = inner;
        swap (six[smallest], six[outer]);
    }
}

```

```

void swap(double &a, double &b)
//Pre:  a and b are defined.
//Post: Values in a and b are swapped.
{
    double temp = a;
    a = b;
    b = temp;
}
double mean (const distance &seven)
//Pre:  Seven is defined and full.
//Post: The mean of the values in seven is returned.
{
    int items = seven.length();
    double sum (0);
    for (int i(0); i < items; i++)
        sum += seven[i];
    return sum/items;
}

double median (const distance &seven)
//Pre:  Seven is defined and full.
//Post: The median of the values in seven is returned.
{
    int items = seven.length();
    int center = items/2;
    return seven[center];
}

void outputData (distance & miles)
//Pre:  Miles is defined and full.
//Post: The data is displayed.
{
    sort (miles);
    cout << "For the given location " << endl;
    cout << " The longest distance is " << miles[miles.length()-1] << endl;
    cout << "The shortest distance is " << miles[0] << endl;
    cout << "      The mean distance is " << mean (miles) << endl;
    cout << "      The median distance is " << median (miles) << endl;
}

```

```

/* cityb.dat    Input data file

```

```

16 15
2 14 H 1 1
1 13 H 1 1
4 12 H 1 1
3 10 H 1 1
14 1 H 1 1
10 10 I 0 6
6 6 S 0 2

```

Output

[illegible]

Enter the x and y coordinates of the proposed location: 8 10

For the given location

The longest distance is 15

The shortest distance is 5

The mean distance is 9.2

The median distance is 10

\* /

```
// Lab 1C
```

```
#include <iostream.h>
```

```
#include <fstream.h>
```

```
#include <apmatrix.h>
```

```
#include <math.h>
```

```
#include <conio.h>
```

```
#include <iomanip.h>
```

```
struct facility
```

```
{
    char type; // H - single-family, A - multi-family, I - factory, S - store
    int num;    // number of families, 0 for factories and stores
    double fireRanking; // Relative measure of fire fighting requirements
    facility():type('.'),num(0),fireRanking(0){};
};
```

```
struct datum
```

```
{
    double mean;
    double median;
};
```

```
typedef apmatrix<facility> city;
```

```
typedef apvector<double> distance;
```

```
typedef apmatrix<datum> data;
```

```
void displayMap (const city &);
```

```
//Pre: City object is defined.
```

```
//Post: Map of facilities and map of population displayed.
```

```
void getMap (city &);
```

```
//Pre: City object is defined.
```

```
//Post: Data in file cityb.dat stored in city object.
```

```

void getLocation (int &, int &, const city &);

//Pre:  City object is defined. ints are declared.
//Post: User inputted location stored in the ints.

void measure(const city &, distance&, int, int);

//Pre:  City object and ints are defined.  Distance object is declared.

//Post: The distance of each dwelling from location passed via ints is
//       determined and stored in distance object.  Multi-family dwellings
//       are weighted according to number of families.

void sort (distance &);

//Pre:  Distance object is defined and full.

//Post: Distance object is sorted in ascending order.

void swap (double &, double &);

//Pre:  The doubles are defined.
//Post: The values in the doubles are swapped.

double mean (const distance &);
//Pre:  Distance object is defined.
//Post: The mean of the elements of distance object is returned.

double median (const distance &);
//Pre:  Distance object is defined.
//Post: The median of the elements of distance object is returned.

void iterate(const city &, data &);

//Pre:  City object is defined and data object is declared.

//Post: Mean distance and median distance for each location in the city
//       object are calculated and stored in data object.

void displayMeans(const data &);

//Pre:  Data object is defined.

//Post: Mean value corresponding to each location is displayed.

void displayMedians(const data &);
//Pre:  Data object is defined.

//Post: Median value corresponding to each location is displayed.

void findSmallest(const data &, city &);

```

```
//Pre: Data and city objects are defined.

//Post: The smallest median is identified. If multiple smallest medians
//       are found, then the median with the smallest mean value is selected
//       and marked on the display.
```

```
int main()
{
    city squareston;

    getMap(squareston);

    displayMap (squareston);

    getch();

    data nums;
    iterate(squareston, nums);
    cout <<"Medians:"<<endl;
    displayMedians(nums);
    getch();
    cout <<"Means:"<<endl;
    displayMeans(nums);
    getch();
    findSmallest(nums, squareston);
    displayMap (squareston);
    getch();
    return 0;
}

void getMap (city &two)
//Pre: City is defined.
//Post: Data in file cityc.dat stored in city.
{
    ifstream inFile ("cityc.dat");
    if (!inFile)
    {
        cerr << "File is not found.";
        exit (-1);
    }
    int rows, cols, r, c;
    inFile >> rows >> cols;
    two.resize (rows, cols);
    while (inFile >> r >> c)
        inFile >> two[r][c].type >> two[r][c].num >> two[r][c].fireRanking;
}

void displayMap (const city &one)
```

```

//Pre: City is defined.

//Post: Map of facilities and map of population displayed.
{
    int rows = one.numrows();

    int cols = one.numcols();
    for(int r = rows -1; r>=0; r--)
    {
        cout <<endl << setw(2) << r;
        for (int c = 0; c<cols; c++)
            cout << setw(2)<<one[r][c].type;
    }
    cout << endl << " ";
    for (int c = 0; c<cols; c++)
        cout << setw(2)<<c/10;
    cout << endl << " ";
    for (int c = 0; c<cols; c++)
        cout << setw(2)<<c%10;
    cout <<endl;
}

void measure (const city &three, distance &four, int x, int y)
//Pre: City, x, and y are defined. Four is declared.
//Post: The distance of each dwelling from x and y is determined and stored
//       in four. Multi-family dwellings are weighted according to number
//       of families.

{
    int rows = three.numrows();
    int cols = three.numcols();
    four.resize (rows*cols);
    int num(0);
    for(int r = 0; r<rows; r++)
    {
        for (int c = 0; c<cols; c++)
            if (three[r][c].type != '.'&&three[r][c].num!= 0)
                for (int i = 0; i<three[r][c].num; i++)
                    {four[num]= fabs(r-x)+fabs(c-y);
                     num++;
                    }
    }
    four.resize(num);
}

void sort(distance &six)
//Pre: Six is defined and full.
//Post: Six is sorted in ascending order.
{
    int smallest, items = six.length();
    for (int outer(0); outer < items - 1; outer++)
    {
        smallest = outer;
        for (int inner = outer + 1; inner < items; inner++)
            if (six[inner] < six[smallest])

```

```

        smallest = inner;
        swap (six[smallest], six[outer]);
    }
}

void swap(double &a, double &b)

//Pre:  a and b are defined.

//Post: Values in a and b are swapped.

{

    double temp = a;

    a = b;

    b = temp;

}


double mean (const distance &seven)

//Pre:  Seven is defined and full.

//Post: The mean of the values in seven is returned.
{
    int items = seven.length();
    double sum (0);
    for (int i(0); i < items; i++)
        sum += seven[i];
    return sum/items;
}


double median (const distance &seven)
//Pre:  Seven is defined and full.
//Post: The median of the values in seven is returned.
{
    int items = seven.length();
    int center = items/2;
    return seven[center];
}


void iterate (const city & grid, data & nums)
//Pre:  Grid is defined and means and medians are declared.
//Post: The mean distance and median distance for each location in
//      grid are calculated and stored in nums.
{

    int rows = grid.numrows();

    int cols = grid.numcols();

    nums.resize(rows, cols);

```

```

    for (int r(0); r<rows; r++)

        for (int c(0); c<cols; c++)

            {

                distance miles;
                measure(grid, miles, r, c);
                sort (miles);
                nums[r][c].mean = mean(miles);
                nums[r][c].median = median(miles);
            }
    }

void displayMeans(const data &nums)
    //Pre: Num is defined.
    //Post: The value corresponding to each location is displayed.
    {
        int rows = nums.numrows();
        int cols = nums.numcols();
        cout << setprecision(3);
        for (int r(rows - 1); r>=0; r--)
            {
                cout <<endl << setw(2) << r;
                for (int c(0); c<cols; c++)
                    cout << setw(5)<<nums[r][c].mean;
            }
        cout << endl <<" ";
        for (int c(0); c<cols; c++)
            cout << setw(5)<< c;
        cout <<endl<<endl;
    }

void displayMedians(const data &nums)
    //Pre: Num is defined.
    //Post: The value corresponding to each location is displayed.
    {
        int rows = nums.numrows();
        int cols = nums.numcols();
        cout << setprecision(3);
        for (int r(rows - 1); r>=0; r--)
            {
                cout <<endl << setw(2) << r;
                for (int c(0); c<cols; c++)
                    cout << setw(5)<<nums[r][c].median;
            }
        cout << endl <<" ";
        for (int c(0); c<cols; c++)
            cout << setw(5)<< c;
        cout <<endl<<endl;
    }

void findSmallest(const data &nums, city & grid)

    //Pre: Num and grid are defined.

```

```

//Post: The smallest median in num is identified. If multiple smallest
medians
//      are found, then the median with the smallest mean value is selected
//      and marked on the display.

{

    int rows = nums.numrows();

    int cols = nums.numcols();

    int r(0), c(0), count(0);

    double smallest = nums[0][0].median;
    for (int row(0); row < rows; row++)
        for (int col(0); col < cols; col++)
            if (nums[row][col].median < smallest)
                { r = row;
                  c = col;
                  smallest = nums[row][col].median;
                }
    double least = nums[r][c].mean;
    for (int row(0); row < rows; row++)
        for (int col(0); col < cols; col++)
            if (nums[row][col].median == smallest && nums[row][col].mean < least)
                { r = row;
                  c = col;
                  least = nums[row][col].mean;
                }

    grid[r][c].type = 'L';
}

```

```

/* cityc.dat  Input data file

```

```

16 15

```

```

13 5 H 1 1

```

```

13 7 A 4 2

```

```

10 3 H 1 1

```

```

9 2 H 1 1

```

```

9 7 A 4 2

```

```

8 5 H 1 1

```

```

8 11 I 0 6

```

```

7 8 S 0 .7

```

```

7 9 S 0 .7

```

7 12 I 0 6

6 3 H 1 1

5 2 H 1 1

5 9 S 0 .7

5 11 I 0 7

4 4 H 1 1

4 6 H 1 1

3 3 H 1 1

2 7 A 6 2

Output

15 . . . . .

14 . . . . .

13 . . . . . H . A . . . . .

12 . . . . .

11 . . . . .

10 . . . H . . . . .

9 . . H . . . A . . . . .

8 . . . . . H . . . . I . . .

7 . . . . . S S . . I . . .

6 . . . H . . . . .

5 . . H . . . . S . I . . .

4 . . . . H . H . . . . .

3 . . . H . . . . .

2 . . . . . A . . . . .

1 . . . . .

0 . . . . .

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4

Medians:

15	13	12	11	10	9	9	10	11	12	13	14	15	16	17	18
14	12	11	10	9	8	8	9	10	11	12	13	14	15	16	17
13	11	10	9	8	7	7	8	9	10	11	12	13	14	15	16
12	10	9	8	7	6	6	7	8	9	10	11	12	13	14	15
11	9	8	7	6	5	5	6	7	8	9	10	11	12	13	14
10	10	9	8	7	6	5	5	6	7	8	9	10	11	12	13
9	9	8	7	6	7	6	5	6	7	8	9	10	11	12	13
8	10	9	8	7	6	6	6	6	7	8	9	10	11	12	13
7	9	8	7	6	5	6	6	5	6	7	8	9	10	11	12
6	10	9	8	7	6	6	5	4	5	6	7	8	9	10	11
5	10	9	8	7	6	5	5	4	5	6	7	8	9	10	11
4	9	8	7	6	5	4	5	5	6	7	8	9	10	11	12
3	8	7	6	7	6	5	6	6	7	8	9	10	11	12	13
2	9	8	7	8	7	6	7	7	8	9	10	11	12	13	14
1	10	9	8	9	8	7	8	8	9	10	11	12	13	14	15
0	11	10	9	10	9	8	9	9	10	11	12	13	14	15	16
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Means:

15	13.7	12.7	11.7	10.8	10.3	9.78	9.48	9.26	10.3	11.3	12.3	13.3	14.3	15.3	16.3
14	12.7	11.7	10.7	9.83	9.26	8.78	8.48	8.26	9.26	10.3	11.3	12.3	13.3	14.3	15.3
13	11.7	10.7	9.65	8.83	8.26	7.78	7.48	7.26	8.26	9.26	10.3	11.3	12.3	13.3	14.3
12	11.1	10.1	9.09	8.26	7.7	7.22	6.91	6.7	7.7	8.7	9.7	10.7	11.7	12.7	13.7
11	10.5	9.52	8.52	7.7	7.13	6.65	6.35	6.13	7.13	8.13	9.13	10.1	11.1	12.1	13.1
10	9.96	8.96	7.96	7.13	6.57	6.09	5.78	5.57	6.57	7.57	8.57	9.57	10.6	11.6	12.6
9	9.48	8.48	7.48	6.65	6.09	5.61	5.3	5.09	6.09	7.09	8.09	9.09	10.1	11.1	12.1
8	9.43	8.43	7.43	6.61	6.04	5.57	5.26	5.04	6.04	7.04	8.04	9.04	10	11	12
7	9.48	8.48	7.48	6.65	6.09	5.61	5.3	5.09	6.09	7.09	8.09	9.09	10.1	11.1	12.1
6	9.52	8.52	7.52	6.7	6.13	5.65	5.35	5.13	6.13	7.13	8.13	9.13	10.1	11.1	12.1
5	9.65	8.65	7.65	6.83	6.26	5.78	5.48	5.26	6.26	7.26	8.26	9.26	10.3	11.3	12.3
4	9.87	8.87	7.87	7.04	6.48	6	5.7	5.48	6.48	7.48	8.48	9.48	10.5	11.5	12.5
3	10.3	9.26	8.26	7.43	6.87	6.39	6.09	5.87	6.87	7.87	8.87	9.87	10.9	11.9	12.9
2	10.7	9.74	8.74	7.91	7.35	6.87	6.57	6.35	7.35	8.35	9.35	10.3	11.3	12.3	13.3
1	11.7	10.7	9.74	8.91	8.35	7.87	7.57	7.35	8.35	9.35	10.3	11.3	12.3	13.3	14.3
0	12.7	11.7	10.7	9.91	9.35	8.87	8.57	8.35	9.35	10.3	11.3	12.3	13.3	14.3	15.3
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```

15 . . . . . . . . . . . . . . .
14 . . . . . . . . . . . . . . .
13 . . . . . H . A . . . . . . .
12 . . . . . . . . . . . . . . .
11 . . . . . . . . . . . . . . .
10 . . . H . . . . . . . . . . .
 9 . . H . . . . A . . . . . . .
 8 . . . . . H . . . . . I . . .
 7 . . . . . . . . . S S . . I . .
 6 . . . H . . . L . . . . . . .
 5 . . H . . . . . . S . I . . .
 4 . . . . H . H . . . . . . . .
 3 . . . H . . . . . . . . . . .
 2 . . . . . . . A . . . . . . .
 1 . . . . . . . . . . . . . . .
 0 . . . . . . . . . . . . . . .
 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
*/

```